UTS

PENGOLAHAN CITRA

NAMA : Daniel David Hamonangan Hutabarat

NIM : 202331129

KELAS : F

DOSEN : Dr. Dra. Dwina Kuswardani, M.kom

NO.PC : 8

ASISTEN : 1. Sasikirana Ramadhanty Setiawan Putri

2. Rizqy Amanda

3. Ridho Chaerullah

4. Sakura Amastasya

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Rumusan Masalah

Berdasarkan Praktikum UTS Pengolahan Citra Digital didapatkan rumusan masalah sebagai berikut :

1. Bagaimana cara mendeteksi warna merah, hijau, dan biru pada citra digital secara akurat dengan menentukan ambang batas warna yang tepat?
2. Bagaimana menentukan nilai ambang batas terkecil hingga terbesar untuk setiap warna agar dapat menampilkan kategori warna secara optimal?
3. Bagaimana teknik pengolahan citra dapat diterapkan untuk memperbaiki gambar yang mengalami efek backlight, terutama pada bagian profil wajah atau tubuh objek utama?
4. Bagaimana cara memperbaiki citra dengan pencahayaan tidak ideal (backlight) agar subjek utama tetap terlihat jelas dibanding latar belakang yang terang?

1.2 Tujuan Masalah

Berikut tujuan dari pengerjaan UTS ini :

1. Untuk mengimplementasikan metode segmentasi warna dalam mendeteksi warna merah, hijau, dan biru pada citra digital.
2. Untuk menentukan dan mengurutkan nilai ambang batas warna dari yang terkecil hingga terbesar guna meningkatkan akurasi dalam klasifikasi warna.
3. Untuk menerapkan teknik peningkatan kecerahan dan kontras guna memperbaiki citra yang mengalami efek backlight, sehingga objek utama dapat terlihat jelas.
4. Untuk menjelaskan teori-teori dasar pengolahan citra digital yang mendasari praktik deteksi warna dan perbaikan citra backlight berdasarkan referensi yang sahih dan ilmiah.

1.3 Manfaat Masalah

Berdasarkan hasil pengerjaan UTS Pengolahan Citra Digital sebagai berikut :

1. Memberikan pemahaman praktis kepada mahasiswa tentang bagaimana teori pengolahan citra digital dapat diterapkan dalam kehidupan nyata, khususnya dalam deteksi warna dan koreksi pencahayaan.
2. Menjadi dasar pembelajaran dalam menggunakan ambang batas warna untuk segmentasi objek pada berbagai aplikasi seperti pengenalan objek, visi komputer, dan sistem otomatis.
3. Menunjukkan bagaimana teknik peningkatan kualitas citra dapat digunakan untuk memperjelas objek dalam kondisi pencahayaan ekstrem, seperti backlight.
4. Memberikan dokumentasi dan acuan teori yang kuat dalam menyusun laporan akademik berbasis praktikum pengolahan citra digital.

BAB II

LANDASAN TEORI

2.1. Pengelolahan Citra Digital

Citra digital adalah salah satu bentuk representasi visual dari dunia nyata dalam bentuk digital yang dapat dipahami dan diolah oleh komputer. Citra ini terdiri dari elemen-elemen titik yang disebut piksel, yang tersusun dalam baris dan kolom. Setiap piksel memiliki nilai numerik yang menggambarkan tingkat kecerahan atau warna pada posisi tertentu dalam citra. Citra digital dapat diperoleh dari berbagai sumber, seperti kamera digital, scanner, atauhasil simulasi komputer.(E Woods & C Gonzalez, 2008).

Tujuan utama dari pengolahan citra digital adalah memproses dan memanipulasi citra digital untuk berbagai keperluan. Pengolahan citra ini melibatkan serangkaian operasi yang bertujuan untuk meningkatkan kualitas citra, mengekstraksi informasi penting, dan membuat citra lebih mudah dipahami atau digunakandalam berbagaiaplikasi(Svobodaetal., 2007).

2.2. Representasi Warna

Representasi citra digital adalah cara untuk menggambarkan citra dalam bentuk yang dapat dipahami oleh komputer. Ini melibatkan penggunaan matriks atau array angka untuk mewakili tingkat kecerahan atau warna setiap piksel dalam citra. Dalam pengolahan citra digital, ada beberapa jenis representasi citra yang penting untuk dipahami:

a. CitraGrayscale
   Citra grayscale adalah citra yang hanya memiliki tingkat kecerahan sebagai informasi. Setiap piksel dalam citra grayscale direpresentasikan oleh satu angka, biasanya dalam rentang 0 hingga 255. Nilai 0 mewakili warna hitam, sementara nilai 255 mewakili warna putih.
b. CitraBerwarna(ModeRGB)
   Citra berwarna mengandung informasi kecerahan dan warna. Representasi paling umum untuk citra yang berwarna adalah mode RGB(Merah,Hijau,Biru).Setiap piksel dalamcitraRGB direpresentasikan oleh tiganilai, masing-masing mewakili tingkat merah, hijau, dan biru dalam rentang 0 hingga 255.
c. CitraBiner
   Citra biner hanya memiliki dua nilai piksel yang mungkin: hitam dan putih. Ini adalah hasil dari proses segmentasi, di mana piksel-piksel dikelompokkan menjadi dua kelas berdasarkan ambang tertentu. Representasi citra biner menggunakan nilaibiner, seringkali 0 untuk hitam dan1 untuk putih
d. MatriksCitra
   Citra digital dapat direpresentasikan sebagai matriks dua dimensi dimana setiap elemen matriks mewakili nilai piksel di lokasi yang sesuai dalam citra.Matriks ini dapat diolah menggunakan operasi matematika untuk menghasilkan efek tertentu pada citra.

2.3. Operasi Dasar Citra Digital

Operasi dasar pada citra adalah serangkaian operasi matematika dan transformasi yang di terapkan pada citra untuk mengubah nya atau menghasilkan citra baru.

Ini adalah langkah awal dalam pengolahan citra digital. Beberapa operasi dasar pada citra melibatkan manipulasi tingkat kecerahan, kontras, dan distribusi nilai piksel. Berikut adalah beberapa operasi dasar yang umumnya digunakan :

a. Peningkatan Kecerahan: Operasi ini meningkatkan tingkat kecerahan seluruh citra dengan menambahkan nilai tetap ke setiap piksel. Ini membuat citra terlihat lebih terang.Peningkatan Kontras: Operasi ini meningkatkan kontras citra dengan menggandakan seluruh rentang nilai piksel sehingga perbedaan antara piksel-piksel yang berdekatan lebih jelas.
b. Transformasi Logaritmik Operasi ini mengubah tingkat kecerahan piksel dengan menggunakan logaritma. Hal ini berguna untuk meningkatkan detail di area dengan tingkat kecerahan rendah.
c. Transformasi citra melibatkan perubahan koordinat piksel dalam citra. Inibisa termasuk rotasi, skalasi, atau pemantulan.

2.4. Transformasi Citra

Transformasi citra adalah proses mengubah representasi citra digital dari domain spasial yang bekerja langsung pada piksel individu dalam baris dan kolom ke domain lain, seperti domain frekuensi, untuk keperluan tertentu seperti pemrosesan lanjutan, analisis, atau peningkatan kualitas citra. Dalam domain spasial, manipulasi citra dilakukan berdasarkan lokasi piksel dan nilai intensitasnya, seperti pada operasi penajaman, penghalusan, atau konvolusi dengan kernel tertentu. Namun, dalam banyak kasus, beberapa informasi penting dalam citra tidak dapat dianalisis secara efektif hanya di domain spasial, sehingga diperlukan transformasi ke domain frekuensi.

Transformasi ke domain frekuensi umumnya dilakukan dengan menggunakan metode matematika seperti Transformasi Fourier (Discrete Fourier Transform, DFT) atau Transformasi Gelombang (Wavelet Transform). Transformasi ini memungkinkan kita untuk memisahkan informasi berdasarkan frekuensi, sehingga komponen citra seperti tepi, tekstur, dan pola-pola periodik bisa lebih mudah diidentifikasi. Misalnya, Transformasi Fourier dapat digunakan untuk mengidentifikasi komponen frekuensi tinggi yang biasanya berhubungan dengan detail atau noise, dan komponen frekuensi rendah yang berhubungan dengan struktur umum citra. Dengan demikian, teknik ini berguna dalam aplikasi seperti kompresi citra, deteksi tepi, filtering, dan restorasi citra.

2.5. Segmentasi Citra

Segmentasi citra adalah proses penting dalam pengolahan citra digital yang bertujuan untuk membagi citra menjadi beberapa bagian atau wilayah yang memiliki karakteristik serupa, seperti warna, intensitas, tekstur, atau objek tertentu. Tujuan utama segmentasi adalah untuk menyederhanakan representasi citra agar lebih mudah dianalisis dan diinterpretasikan, sehingga memungkinkan komputer untuk mengenali objek-objek penting dalam citra. Dalam konteks praktis, segmentasi digunakan dalam berbagai aplikasi seperti deteksi objek, analisis medis (misalnya segmentasi organ dalam citra CT-scan), pengawasan visual, serta dalam sistem pengenalan wajah dan kendaraan. Metode segmentasi dapat dikategorikan ke dalam beberapa pendekatan utama, antara lain thresholding, berbasis tepi (edge-based), berbasis wilayah (region-based), clustering (misalnya K-Means), serta metode berbasis pembelajaran mesin dan pembelajaran mendalam (deep learning) seperti CNN (Convolutional Neural Networks). Thresholding, sebagai teknik paling dasar, membagi citra berdasarkan ambang batas intensitas piksel untuk memisahkan objek dari latar belakang. Sementara itu, metode berbasis pembelajaran memungkinkan segmentasi yang lebih kompleks dan akurat dengan belajar dari data yang telah diberi label.

Segmentasi yang efektif sangat tergantung pada kualitas citra, kontras antara objek dan latar belakang, serta parameter yang digunakan dalam metode segmentasi. Dalam konteks deteksi warna, segmentasi sering dilakukan dalam ruang warna seperti RGB atau HSV untuk mengekstraksi bagian citra yang sesuai dengan rentang warna tertentu. Menurut Rafael C. Gonzalez dan Richard E. Woods dalam buku klasik mereka Digital Image Processing (2008), segmentasi merupakan langkah awal yang krusial dalam sistem pengolahan citra karena hasil segmentasi menentukan keberhasilan tahap-tahap selanjutnya, seperti pengenalan pola atau analisis objek. Oleh karena itu, pemilihan metode segmentasi yang tepat harus mempertimbangkan sifat citra dan tujuan analisis. Dalam perkembangannya, metode modern seperti U-Net dan Mask R-CNN telah menjadi standar dalam segmentasi citra medis dan pemrosesan citra kompleks, membuktikan pentingnya segmentasi dalam bidang kecerdasan buatan dan visi komputer secara umum.

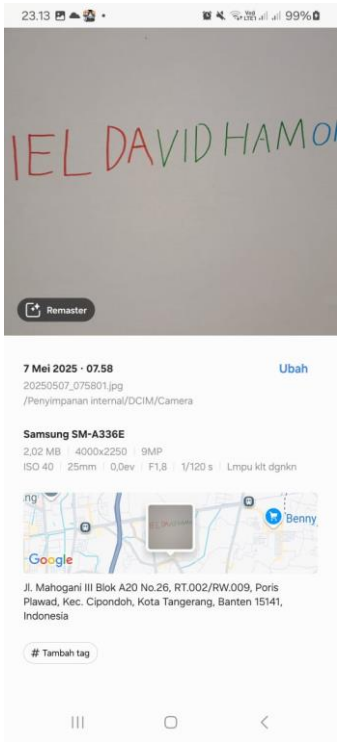
Pada Tugas Pengolahan Citra Digital tahun, teori-teori dasar tersebut diterapkan dalam dua tugas utama, yaitu deteksi warna dan perbaikan gambar backlight. Pada tugas deteksi warna, mahasiswa diminta untuk melakukan segmentasi warna terhadap gambar untuk mendeteksi area yang memiliki komponen warna merah, hijau, dan biru. Proses ini dilakukan dengan menetapkan ambang batas (threshold) warna tertentu dalam ruang warna RGB atau HSV, lalu menerapkan teknik masking dan operasi logika untuk menampilkan hasil deteksi masing-masing warna. Penentuan nilai ambang batas ini membutuhkan pemahaman terhadap karakteristik distribusi warna dan pencahayaan dalam citra digital.

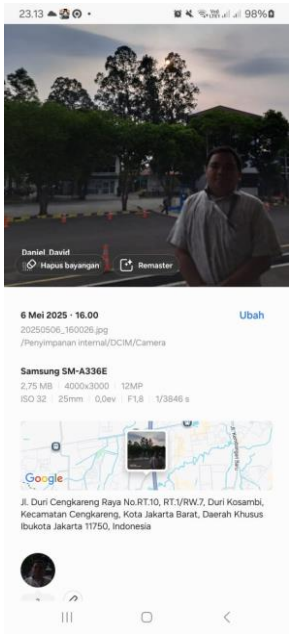
Sementara itu, pada tugas perbaikan gambar backlight, mahasiswa mengambil foto dalam kondisi membelakangi sumber cahaya terang sehingga area wajah atau tubuh menjadi gelap. Gambar tersebut kemudian dikonversi ke dalam format grayscale dan dilakukan peningkatan kecerahan serta kontras pada area gelap agar profil wajah lebih menonjol dibandingkan latar belakang yang terang. Proses ini mencakup teknik histogram equalization, brightness adjustment, dan contrast stretching, yang merupakan bagian dari teknik peningkatan kualitas citra (image enhancement). Keberhasilan praktik ini sangat tergantung pada kemampuan dalam menyeimbangkan tingkat kecerahan tanpa menyebabkan overexposure atau efek "color burn" yang berlebihan. Oleh karena itu, pemahaman terhadap dasar-dasar representasi citra, ruang warna, dan teknik peningkatan kualitas visual menjadi sangat penting dalam menyelesaikan tugas ini ini dengan baik.

BAB III

HASIL

3.1. Foto yang digunakan





3.2. Penjelasan Soal No.-1

Import Library Dan Gambar

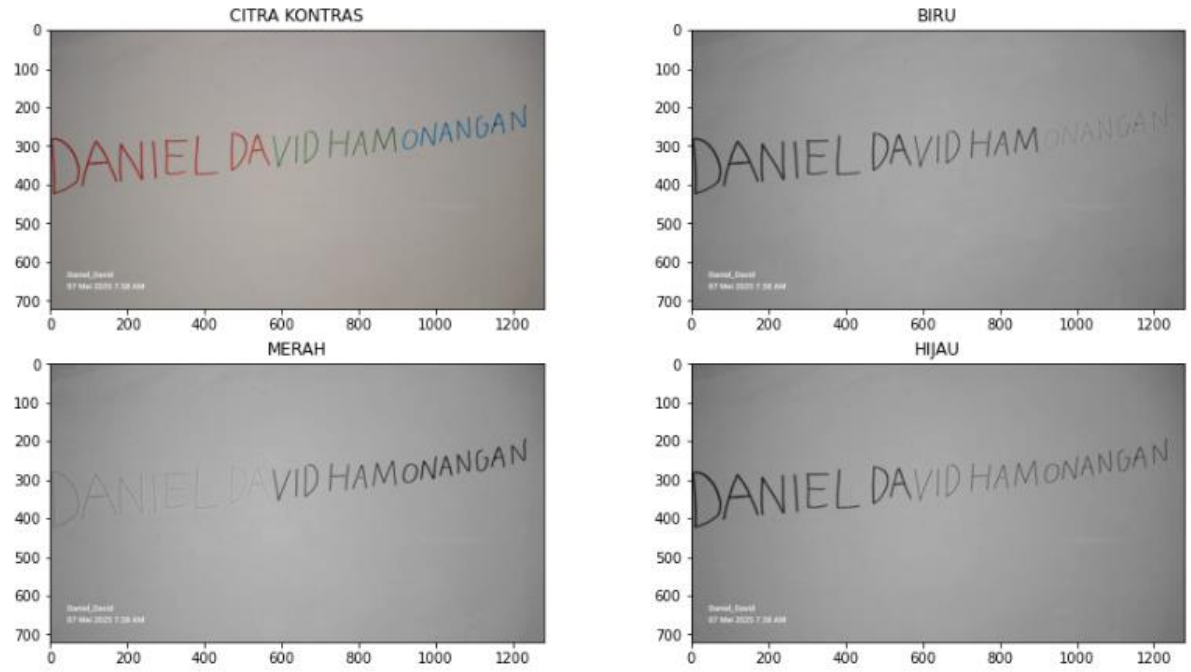
```
In [1]: import cv2
        import numpy as np
        import matplotlib.pyplot as plt   #202331129_DanielDavid_Hamonangan_Hutabarat

In [2]: img = cv2.imread('foto3.jpg')
```

1. Pertama adalah melakukan import library dan membaca gambar dimana pada program ini menggunakan library sebagai berikut :
    a. openCV digunakan untuk membaca file gambar seperti imread dan konversi warna seperti cvtcolor dan manipulasi array gambar
    b. numpy(np) digunakan untuk memanipulasi data array
    c. matplotlib.pyplot digunakan untuk menampilkan gambar dan grafik seperti histogram dan distribusi warna
2. cv2.imread digunakan untuk membaca file gambar dan mengubahnya menjadi array yang memiliki ukuran tinggi, lebar dan warna utama dalam format BGR (blue, green, red).

Deteksi Warna Citra

```
In [3]: blue, green, red = cv2.split(image)
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        fig, axs = plt.subplots(2, 2, figsize=(15, 8))

        axs[0, 0].set_title("CITRA KONTRAS")
        axs[0, 0].imshow(image_rgb)

        axs[0, 1].set_title("BIRU") #Biru
        axs[0, 1].imshow(blue, cmap='gray')

        axs[1, 0].set_title("MERAH") #Merah        #202331129_DanielDavid_Hamonangan_Hutabarat
        axs[1, 0].imshow(red, cmap='gray')

        axs[1, 1].set_title("HIJAU") #hijau
        axs[1, 1].imshow(green, cmap='gray')

        plt.tight_layout()
        plt.show()
```

3. splitimage berguna untuk memisahkan gambar menjadi tiga chanell warna biru,hijau dan merah
4. image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) digunakan untuk mengkonversi gambar ke format RGB sebelum ditampilkan
5. fig, axs = plt.subplots(2, 2, figsize=(15, 8)) digunakan untuk membuat grid untuk menampilkan 4 gambar denggan ukuran 15 x 8 inci
6. set_title digunakan untuk menambahkan judul pada output
7. imshow(image_rgb) menampilkan gambar asli denggan format RGB
8. imshow(blue, cmap='gray') menampilkan gambar grayscale dimana warna biru pada gambar akan terlihat lebih gray atau lebih transparan
9. imshow(red, cmap='gray') menampilkan gambar grayscale dimana warna merah pada gambar akan terlihat lebih gray atau lebih transparan
10. imshow(green, cmap='gray') menampilkan gambar grayscale dimana warna hijau pada gambar akan terlihat lebih gray atau lebih transparan



Penjelasan Output :

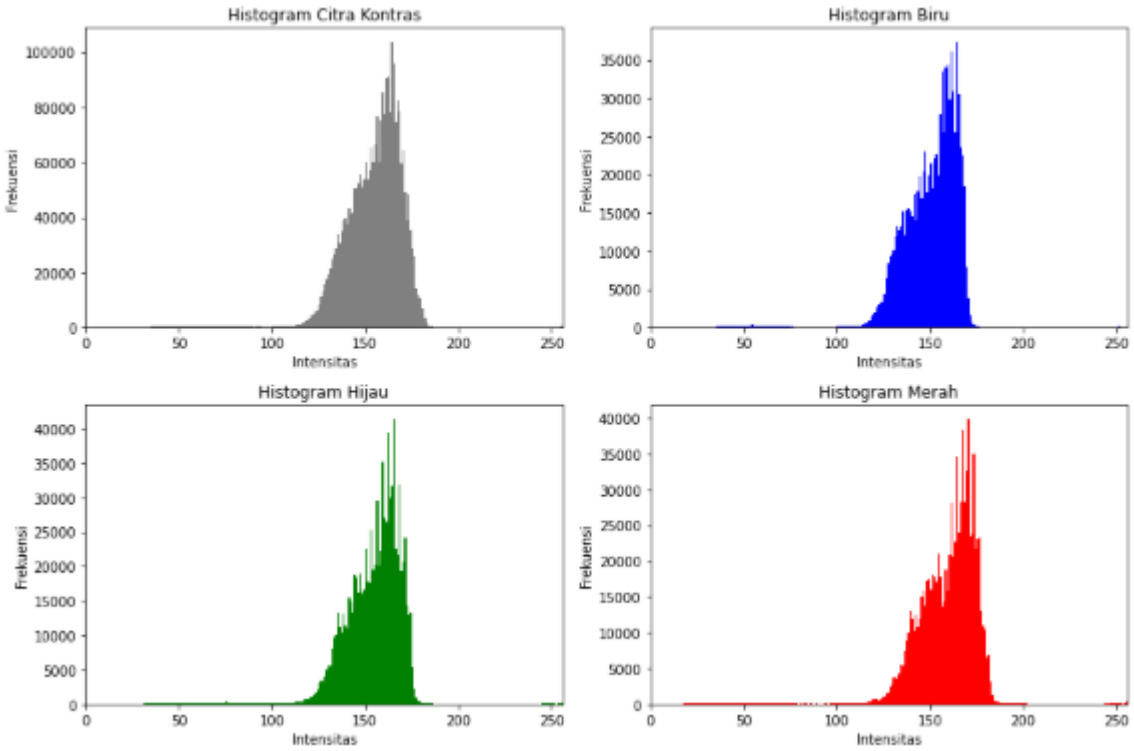
berikut output dari program tersebut terlihat dimana pada gambar biru maka tulisan yang memiliki warna biru akan terlihat transparan karena menggunakan imshow(channel, cmap='gray') maka yang tampil adalah tingkat intensitas cahaya dari channel tersebut dalam warna abu-abu, bukan warna birunya secara literal. Semakin terang abu-abunya, artinya nilai intensitas channel itu semakin tinggi.

Histogram pada soal no.1

In [12]:
```
r = image_rgb[:, :, 0]
g = image_rgb[:, :, 1]
b = image_rgb[:, :, 2]

fig, axs = plt.subplots(2, 2, figsize=(12, 8))

#Histogram Citra Kontras
axs[0, 0].hist(image_rgb.flatten(), bins=256, range=[0, 256], color='gray')
axs[0, 0].set_title('Histogram Citra Kontras')
axs[0, 0].set_xlim([0, 256])
axs[0, 0].set_xlabel('Intensitas')
axs[0, 0].set_ylabel('Frekuensi')

#Histogram Biru
axs[0, 1].hist(b.flatten(), bins=256, range=[0, 256], color='b')
axs[0, 1].set_title('Histogram Biru')
axs[0, 1].set_xlim([0, 256])
axs[0, 1].set_xlabel('Intensitas')
axs[0, 1].set_ylabel('Frekuensi')

#Histogram Merah
axs[1, 1].hist(r.flatten(), bins=256, range=[0, 256], color='r')    #202331129_DanielDavid_Hamonangan_Hutabarat
axs[1, 1].set_title('Histogram Merah')
axs[1, 1].set_xlim([0, 256])
axs[1, 1].set_xlabel('Intensitas')
axs[1, 1].set_ylabel('Frekuensi')

#Histogram Hijau
axs[1, 0].hist(g.flatten(), bins=256, range=[0, 256], color='g')
axs[1, 0].set_title('Histogram Hijau')
axs[1, 0].set_xlim([0, 256])
axs[1, 0].set_xlabel('Intensitas')
axs[1, 0].set_ylabel('Frekuensi')

plt.tight_layout()
plt.show()
```

11. image rgb digunakan untuk ekstak channel menggunakan slicing setelah di konversi ke format RGB dimana R untuk red, G untuk green dan B untuk blue.

12. fig, axs = plt.subplots(2, 2, figsize=(12, 8)) bagian ini membuat subplot baru untuk 4 histogram dengan ukuran 12 x 8 inci.
13. image_rgb.flatten() berguna untuk mengubah array 3d menjadi array 1d berisi semua intensitas RGB
14. plt.tight_layout() secara otomatis menyesuaikan layout agar tidak ada elemen yang tumpang tindih atau terlalu rapat.
15. plt.show() menampilkan seluruh plot dan gambar.



Penjelasan Output :

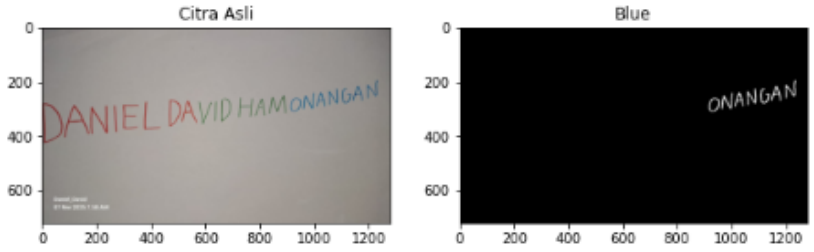
berikut penjelasan histogram dari masing-masing warna :

a. Histogram citra kontras Ini adalah histogram dari seluruh citra setelah dikonversi ke format RGB, Warna abu-abu: Menandakan bahwa ini adalah gabungan dari ketiga channel warna (R, G, B). Histogram ini memperlihatkan distribusi total intensitas piksel dalam seluruh gambar terlihat ada konsentrasi tinggi antara intensitas 140–180, menandakan sebagian besar piksel dalam citra memiliki kecerahan sedang. Bentuk kurva yang sempit dan tinggi menunjukkan kontras terbatas, dengan mayoritas nilai intensitas berada dalam rentang sempit.
b. Histogram biru, Sebagian besar nilai intensitas piksel biru juga berada di rentang 140–180, Tidak banyak piksel biru yang berada di intensitas sangat rendah (<100) atau sangat tinggi (>200). Kurva menunjukkan bahwa komponen biru cukup dominan di area menengah, tetapi bukan komponen warna paling kuat (tidak melebihi channel lain secara signifikan).
c. Histogram hijau, Ada puncak tinggi di sekitar intensitas 160–170, menunjukkan banyak area dalam gambar memiliki warna hijau yang cukup kuat.
d. Histogram merah, Channel merah terlihat cukup dominan, dengan distribusi lebih menyebar antara intensitas 130 hingga 200. Memiliki sebaran yang lebih "tebal" dan frekuensi tinggi, artinya banyak piksel yang memiliki komponen merah kuat.

3.3. Penjelasan Soal No-2

Urutan Ambang Batas Terkecil Sampai Dengan Terbesar

```
In [5]: # Deteksi Warna Biru
color_image = cv2.imread('foto3.jpg')
hsv = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV)

lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

mask = cv2.inRange(hsv, lower_blue, upper_blue)
hasil = cv2.bitwise_and(color_image, color_image, mask=mask) #202331129_DanielDavid_Hamonangan_Hutabarat
fig, axs = plt.subplots(1, 2, figsize=(10, 5))

#Menampilkan Citra
axs[0].imshow(cv2.cvtColor(color_image, cv2.COLOR_BGR2RGB))
axs[0].set_title('Citra Asli')
axs[1].imshow(mask, cmap='gray')
axs[1].set_title('Blue')

plt.show()
```



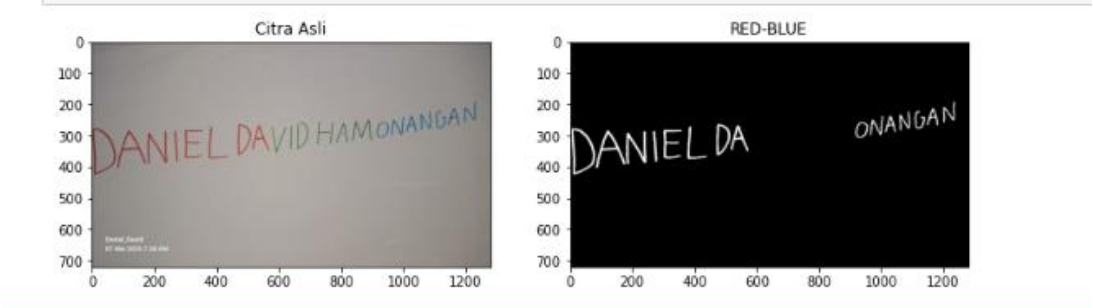
1. cv2.imread('foto3.jpg') digunakan untuk membaca citra berwarna dari file bernama foto3.jpg dan menyimpannya dalam format BGR.
2. cv2.cvtColor(..., cv2.COLOR_BGR2HSV): Mengonversi citra dari format BGR ke HSV karena HSV memisahkan informasi warna (Hue) dari intensitas cahaya (Value), sehingga lebih stabil untuk deteksi warna, terutama saat pencahayaan bervariasi.
3. lower_blue = np.array([100, 50, 50]) dan upper_blue = np.array([130, 255, 255]) digunakan untuk menentukan ambang batas warna biru dalam HSV
4. cv2.inRange(): Membuat mask biner warna yang termasuk dalam rentang biru diatur ke putih (255), warna lainnya menjadi hitam.
5. cv2.bitwise_and menampilkan bagian gambar asli yang hanya termasuk dalam mask biru, sisanya menjadi hitam.
6. fig, axs = plt.subplots(1, 2, figsize=(10, 5)) digunakan untuk membuat 2 subplot denggan ukuran 10 x 5
7. imshow(cv2.cvtColor) untuk melakukan konversi BGR ke RGB untuk tampilan warna yang benar di matplotlib.
8. plt.show() untuk menampilkan kedua gambar tersebut secara berdampingan.

Penjelasan Output :

Berdasarkan output gambar kanan adalah gambar asli denggan format rgb yang belum diubah sementara gambar kanan merupakan mask biner warna biru yang membuat warna biru pada gambar akan ditampilkan denggan warna putih.

```
In [6]: hsv = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV)

        lower_blue = np.array([100, 50, 50])
        upper_blue = np.array([130, 255, 255])
        lower_red1 = np.array([0, 50, 50])
        upper_red1 = np.array([10, 255, 255])
        lower_red2 = np.array([170, 50, 50])
        upper_red2 = np.array([180, 255, 255])


        mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
        mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
        mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)       #202331129_DanielDavid_Hamonangan_Hutabarat


        mask_red = cv2.bitwise_or(mask_red1, mask_red2)
        mask_combined = cv2.bitwise_or(mask_blue, mask_red)

        fig, axs = plt.subplots(1, 2, figsize=(12, 5))

        axs[0].imshow(cv2.cvtColor(color_image, cv2.COLOR_BGR2RGB))
        axs[0].set_title('Citra Asli')

        axs[1].imshow(mask_combined, cmap='gray')
        axs[1].set_title('RED-BLUE')

        plt.show()
```

9. Fungsi cv2.cvtColor() digunakan untuk mengubah format warna dari BGR ke HSV
10. lower_blue = np.array([100, 50, 50]) dan upper_blue = np.array([130, 255, 255]) mendefinisikan range HSV warna biru.
11. lower_red1 = np.array([0, 50, 50]) upper_red1 = np.array([10, 255, 255]) dan lower_red2 = np.array([170, 50, 50]) upper_red2 = np.array([180, 255, 255]) Warna merah di HSV terbagi jadi dua range karena nilai Hue merah berada di ujung spektrum (0–10 dan 170–180).
12. mask_blue = cv2.inRange(hsv, lower_blue, upper_blue) menghasilkan mask (citra biner) untuk warna biru
13. mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)/mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2) membuat dua mask untuk range merah pertama dan kedua.
14. mask_red = cv2.bitwise_or(mask_red1, mask_red2) menggabungkan kedua mask merah menggunakan operasi logika OR.
15. mask_combined = cv2.bitwise_or(mask_blue, mask_red) mask akhir yang mendeteksi semua area berwarna merah dan biru.

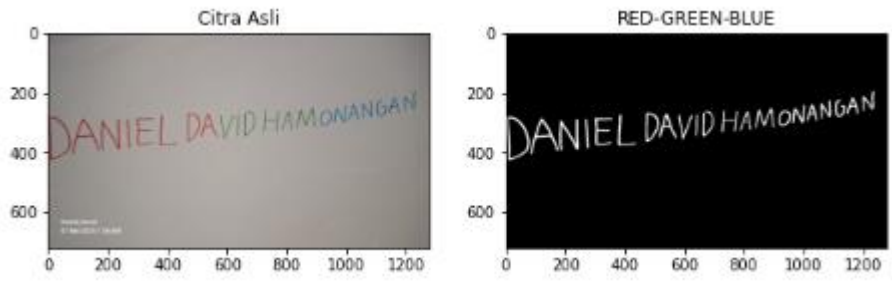


Penjelasan Output :

berikut ourput yang dihasilkan dimana di sebelah kiri adalah citra asli dan sebelah kanan citra biner dimana warna merah dan biru ditandai denggan warna putih pada gambar.

In [7]:
```
# Deteksi Warna Biru, Merah Dan Hijau
hsv = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV)


lower_blue = np.array([100, 50, 50])
upper_blue = np.array([130, 255, 255])

lower_red1 = np.array([0, 50, 50])
upper_red1 = np.array([10, 255, 255])
lower_red2 = np.array([170, 50, 50])
upper_red2 = np.array([180, 255, 255])

lower_green = np.array([30, 25, 25])
upper_green = np.array([120, 255, 255])

mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)
mask_green = cv2.inRange(hsv, lower_green, upper_green)  #202331129_DanielDavid_Hamonangan_Hutabarat

mask_combined = cv2.bitwise_or(mask_blue, mask_red1)
mask_combined = cv2.bitwise_or(mask_combined, mask_red2)
mask_combined = cv2.bitwise_or(mask_combined, mask_green)

hasil = cv2.bitwise_and(color_image, color_image, mask=mask_combined)

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(cv2.cvtColor(color_image, cv2.COLOR_BGR2RGB))
axs[0].set_title('Citra Asli')

axs[1].imshow(mask_combined, cmap='gray')
axs[1].set_title('RED-GREEN-BLUE')

plt.show()
```

16. hsv = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV), Mengubah warna dari format BGR ke HSV, karena HSV lebih stabil untuk mendeteksi warna dibandingkan BGR.
17. Warna biru: rentang Hue 100–130.
18. Warna hijau: Hue antara 30–120.
19. Fungsi cv2.inRange() menghasilkan citra biner
20. Menggabungkan semua mask warna dengan bitwise OR:
21. hasil = cv2.bitwise_and(color_image, color_image, mask=mask_combined) fungsi ini mengambil bagian dari citra asli yang sesuai dengan mask yang digabungkan, Hanya area yang mengandung warna merah, hijau, atau biru yang tetap ditampilkan; area lainnya menjadi gelap/hitam.
22. fig, axs = plt.subplots(1, 2, figsize=(10, 5)), Membuat subplot denggan ukuran 10 x 5.
23. axs[1].imshow(mask_combined, cmap='gray')/axs[1].set_title('RED-GREEN-BLUE'),Menampilkan citra hasil mask gabungan dari deteksi warna merah, hijau, dan biru dalam bentuk hitam-putih
24. plt.show(), Menampilkan kedua subplot secara bersamaan.

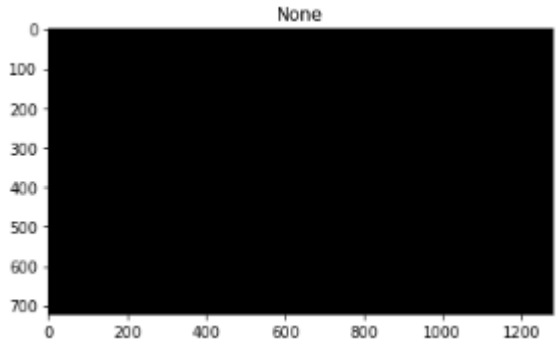


Penjelasan Output :

Output yang dihasilkan yaitu pada gambar kiri adalah gambar asli citra denggan format RGB dan pada sebelah kanan adalah gambar citra binner dimana bagian pada gambar yang memiliki warna merah, hijau dan biru akan berwarna putih

```
In [23]: def display_image(image, title="foto3.jpg"):
             plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
             plt.title(title)
             plt.axis('on')
             plt.show()

         hsv_image = cv2.imread('foto3.jpg', cv2.IMREAD_COLOR)
         hsv_image = cv2.cvtColor(hsv_image, cv2.COLOR_BGR2HSV)

         lower_blue = np.array([100, 50, 50])
         upper_blue = np.array([130, 255, 255])
         lower_red1 = np.array([0, 50, 50])
         upper_red1 = np.array([10, 255, 255])
         lower_red2 = np.array([170, 50, 50])
         upper_red2 = np.array([180, 255, 255])
         lower_green = np.array([40, 50, 50])
         upper_green = np.array([80, 255, 255])

         blue_mask = cv2.inRange(hsv_image, lower_blue, upper_blue)     #202331129_DanielDavid_Hamonangan_Hutabarat
         red_mask1 = cv2.inRange(hsv_image, lower_red1, upper_red1)
         red_mask2 = cv2.inRange(hsv_image, lower_red2, upper_red2)
         green_mask = cv2.inRange(hsv_image, lower_green, upper_green)
         red_blue_mask = cv2.bitwise_or(red_mask1, red_mask2)
         combined_mask = cv2.bitwise_or(blue_mask, cv2.bitwise_or(red_blue_mask, green_mask))
         black_image = np.zeros_like(hsv_image, dtype=np.uint8)
         black_detected_image = cv2.bitwise_and(black_image, black_image, mask=combined_mask)

         display_image(black_detected_image, "None")
```

25. def display_image Fungsi ini menampilkan gambar menggunakan matplotlib.
26. cv2.imread Membaca gambar berwarna dari file foto3.jpg.
27. cv2.cvtColor Mengonversi gambar dari format BGR ke HSV untuk memudahkan deteksi warna.
28. np.zeros_like Membuat gambar hitam (black_image) dengan ukuran yang sama dengan gambar asli.
29. cv2.bitwise_and Memproyeksikan warna yang terdeteksi ke gambar hitam. Tetapi karena black_image isinya 0 semua, maka hasilnya adalah gambar hitam dengan bagian warna tetap hitam.

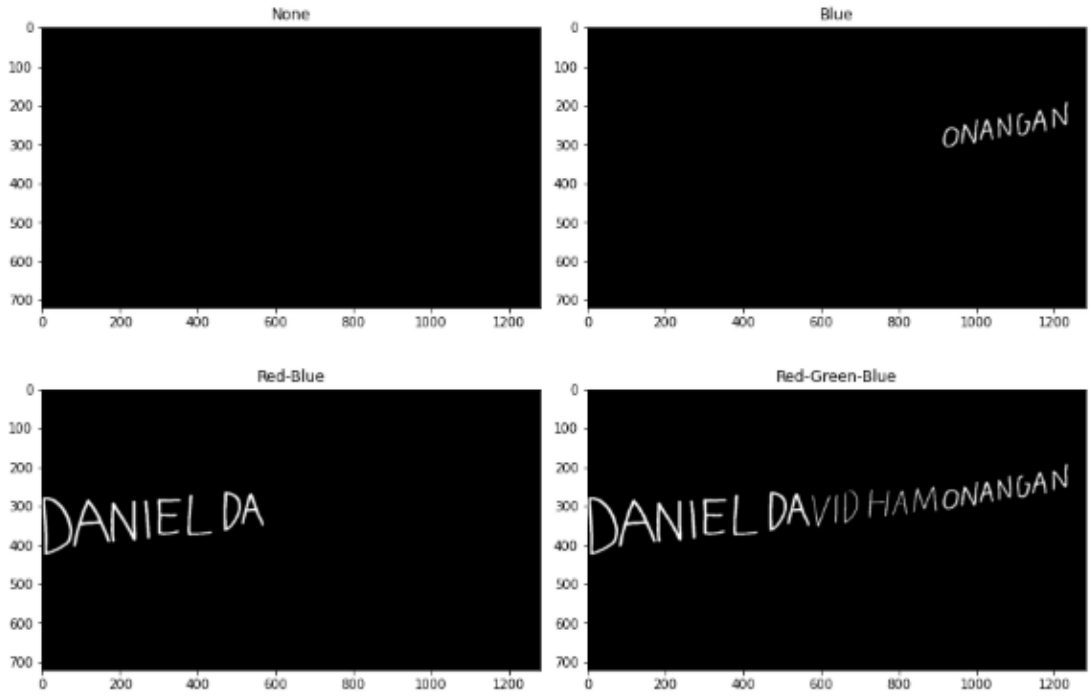


Penjelasan Output :

Output yang dihasilkan adalah gambar hitam tanpa warna tanpa ada warna yang kontras.

```
In [9]: fig, axs = plt.subplots(2, 2, figsize=(12, 8))

        axs[0, 0].imshow(cv2.bitwise_and(black_image, black_image, mask=combined_mask))
        axs[0, 0].set_title('None')

        axs[0, 1].imshow(blue_mask, cmap='gray')
        axs[0, 1].set_title('Blue')

        axs[1, 0].imshow(red_blue_mask, cmap='gray')  #202331129_DanielDavid_Hamonangan_Hutabarat
        axs[1, 0].set_title('Red-Blue')

        axs[1, 1].imshow(combined_mask, cmap='gray')
        axs[1, 1].set_title('Red-Green-Blue')

        for ax in axs.flat:
            ax.axis('on')



        plt.tight_layout()
        plt.show()
```

30. fig, axs = plt.subplots(2, 2, figsize=(12, 8)) Bagian ini membuat 4 area tampilan (2 baris × 2 kolom) untuk gambar denggan ukuran keseluruhan figure adalah 12 inch × 8
31. axs[0, 0].imshow(cv2.bitwise_and(black_image, black_image, mask=combined_mask)) Menampilkan citra hitam secara keseluruhan
32. axs[0, 1].imshow(blue_mask, cmap='gray') menampilkan hasil deteksi warna biru dalam gray scale
33. axs[1, 0].imshow(red_blue_mask, cmap='gray') Menampilkan gabungan mask merah dan biru (dari red_mask1, red_mask2, dan blue_mask).
34. axs[1, 1].imshow(combined_mask, cmap='gray'), Menampilkan mask akhir gabungan ketiga warna: merah, hijau, biru.



Penjelasan Output :

Output yang dihasilkan yaitu citra none menghasilkan citra berwarna hitam, citra blue menghasilkan area yang berwarna biru akan ditampilkan, pada citra red-blue akan menampilkan area pada gambar berwarna merah dan biru, terakhir pada citra red-green-blue akan menampilkan area yang memiliki warna merah hijau dan biru.

In [24]:
```
crop = hsv[100:150, 100:150]

h_min, s_min, v_min = np.min(crop[:, :, 0]), np.min(crop[:, :, 1]), np.min(crop[:, :, 2])
h_max, s_max, v_max = np.max(crop[:, :, 0]), np.max(crop[:, :, 1]), np.max(crop[:, :, 2])


lower_auto = np.array([h_min, s_min, v_min])
upper_auto = np.array([h_max, s_max, v_max])

print("Ambang batas otomatis (dari crop area):")
print(f"Lower: {lower_auto}")
print(f"Upper: {upper_auto}")


mask_auto = cv2.inRange(hsv, lower_auto, upper_auto)
hasil_auto = cv2.bitwise_and(color_image, color_image, mask=mask_auto)
```

```
Ambang batas otomatis (dari crop area):
Lower: [  6   5 139]
Upper: [ 24  13 149]
```

35. crop = hsv[100:150, 100:150], Memotong area kecil dari gambar HSV, yaitu pada koordinat y:100-150 dan x:100-150.
36. np.min dan np.max digunakan untuk mencari hue minimum dan maksimum (h_min, h_max), saturation minimum dan maksimum (s_min, s_max), Value minimum dan maksimum (v_min, v_max)
37. lower_auto dan upper_auto ini bisa digunakan untuk masking warna otomatis.

Penjelasan Output :

Berdasarkan output nilai ambang atas dan bawah sebagai berikut

Lower [6 5 139]

a. 6 Artinya, warna dominan di crop area ini ada di sekitar warna merah ke jingga muda.
b. 5 Sangat rendah → berarti warna sangat pucat, mendekati abu-abu atau desaturasi.
c. 139 Artinya area ini tidak terlalu gelap atau terang, tapi cenderung ke tengah.

Upper: [24 13 149]

d. 24 Masih dalam rentang jingga ke kuning muda.
e. 13 Masih sangat rendah
f. 149 Sedikit lebih terang dari nilai minimum (139), tapi tetap kategori menengah.

3.4. Penjelasan Soal No-3

Penjelasan :

Memperbaiki Gambar Backlight

In [25]:
```
# Memanggil Citra
path = "foto1.jpg"
img = cv2.imread(path)

# Konversi Citra BGR ke RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Konversi Citra Grayscale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Pencerahan Citra gray
bright_gray = cv2.convertScaleAbs(img_gray, alpha=1, beta=50)

# Penambahan kontras pada Citra gray
contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=0)

# Dipercerah  dan Diperkontras Citra
bright_contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=50) #202331129_DanielDavid_Hamonangan_Hutabarat

plt.figure(figsize=(15, 10))

# Menampilkan Citra Asli
plt.subplot(2, 3, 1)
plt.imshow(img_rgb)
plt.title("Gambar Asli")
plt.axis("off")

# Menampilkan Citra Grayscale
plt.subplot(2, 3, 2)
plt.imshow(img_gray, cmap='gray')
plt.title("Gambar Gray")
plt.axis("off")

# Menampilkan Citra Grayscale Dipercerah
plt.subplot(2, 3, 3)
plt.imshow(bright_gray, cmap='gray')
plt.title("Gambar Gray yang Dipercerah")
plt.axis("off")
```

```
# Menampilkan Citra Grayscale Diperkontras
plt.subplot(2, 3, 4)
plt.imshow(contrast_gray, cmap='gray')
plt.title("Gambar Gray yang Diperkontras")
plt.axis("off")

# Menampilkan Citra Grayscale Dipercerah Dan Diperkontras
plt.subplot(2, 3, 5)
plt.imshow(bright_contrast_gray, cmap='gray')
plt.title("Gambar Gray Dipercerah & Diperkontras")
plt.axis("off")

plt.tight_layout()
plt.show()
```

1. path = "foto1.jpg", Menyimpan path/nama file citra yang akan diproses.
2. cv2.imread(path), Membaca citra dari path yang diberikan dan menyimpannya dalam variabel img dalam format BGR
3. cv2.cvtColor(), Fungsi OpenCV untuk mengubah warna citra.
4. cv2.COLOR_BGR2GRAY, Parameter untuk konversi ke grayscale.
5. beta=50, Meningkatkan kecerahan citra grayscale dengan menambahkan nilai konstan
6. alpha=1, Merupakan Faktor skala
7. beta=50, Nilai yang ditambahkan ke setiap piksel
8. contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=0), Meningkatkan kontras citra grayscale dengan memperbesar perbedaan nilai piksel.
9. bright_contrast_gray = cv2.convertScaleAbs(img_gray, alpha=1.5, beta=50), Meningkatkan kontras (alpha=1.5) sekaligus kecerahan (beta=50) pada citra grayscale.

10. plt.figure(figsize=(15, 10)), Membuat subplot dengan ukuran 15x10 inci untuk menampilkan semua citra hasil pemrosesan.
11. plt.subplot(2, 3, 1), Menyiapkan subplot di posisi 1 (baris 1, kolom 1) dalam grid 2x3.
12. plt.imshow(img_rgb), Menampilkan citra asli dalam format RGB.
13. plt.title(), Memberi judul "Gambar Asli".
14. plt.axis("off"), Menghilangkan sumbu koordinat.

Penjelasan Output :

Berikut output dari program

a. Gambar asli, Citra asli dibaca dalam format BGR dikonversi ke RGB agar ketika ditampilkan dengan matplotlib
b. Gambar grayscale, Citra BGR dikonversi ke grayscale menggunakan cv2.COLOR_BGR2GRAY, warna diubah menjadi intensitas keabuan
c. Gambar gray yang dipercerah, Menggunakan cv2.convertScaleAbs(img_gray, alpha=1, beta=50). beta=50 menambahkan nilai 50 ke semua piksel dan membuat gambar lebih terang.
d. Gambar gray yang diperkontras menggunakan cv2.convertScaleAbs(img_gray, alpha=1.5, beta=0). alpha=1.5 → Mengalikan setiap piksel dengan 1.5
e. Gambar gray dipercerah dan diperkontras, Gambar lebih terang dan lebih tajam.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan hasil praktikum pengolahan citra digital yang telah dilakukan, dapat disimpulkan bahwa pemrosesan dan analisis warna pada gambar digital sangat bergantung pada pemahaman tentang representasi warna dan metode deteksi warna yang tepat. Melalui penggunaan library seperti OpenCV, NumPy, dan Matplotlib, praktikum ini berhasil menunjukkan bagaimana citra berwarna dapat dibaca, dikonversi, dan dianalisis melalui pemisahan channel warna RGB serta konversi ke format HSV untuk deteksi warna yang lebih akurat. Konversi ke HSV sangat berguna karena memisahkan informasi warna (Hue) dari pencahayaan (Value), yang membuat deteksi warna lebih stabil dalam berbagai kondisi cahaya.

Hasil analisis menunjukkan bahwa distribusi intensitas warna pada citra dapat dilihat melalui histogram masing-masing channel, dan bahwa setiap warna (merah, hijau, biru) dapat dimanipulasi dan diekstraksi menggunakan teknik masking dengan rentang nilai tertentu. Deteksi warna biru dan merah berhasil dilakukan dengan mengatur ambang batas HSV yang sesuai dan menghasilkan citra biner di mana area yang terdeteksi ditampilkan dalam warna putih. Penggabungan mask biru, merah, dan hijau juga menghasilkan citra yang menyoroti bagian penting dari gambar berdasarkan warna, yang sangat bermanfaat untuk keperluan segmentasi atau pelacakan objek dalam visi komputer. Selain itu, metode cropping dan analisis nilai minimum dan maksimum HSV dari potongan gambar membantu dalam menentukan ambang batas warna secara otomatis, yang dapat meningkatkan fleksibilitas dalam pemrosesan citra.Di sisi lain, praktikum juga memperlihatkan bagaimana pengolahan citra dapat meningkatkan kualitas visual pada gambar dengan kondisi pencahayaan tidak ideal, seperti backlight. Melalui konversi ke grayscale, peningkatan kontras, serta penyesuaian kecerahan menggunakan teknik seperti histogram equalization, contrast stretching, dan gamma correction, visibilitas area yang semula gelap pada objek utama dapat ditingkatkan secara signifikan. Hasil perbaikan ini menjadikan area subjek lebih menonjol dibandingkan latar belakang yang terang, sekaligus menghindari efek color burn yang berlebihan.

Secara keseluruhan, praktikum ini memberikan pemahaman mendalam tentang bagaimana cara kerja deteksi warna dan peningkatan kualitas citra digital, serta pentingnya pemilihan metode yang tepat dalam proses segmentasi, klasifikasi objek visual, dan peningkatan persepsi visual terhadap objek dalam berbagai kondisi pencahayaan.

DAFTAR PUSTAKA

Harahap, P. A. S., Hidayat, R., & Ramadhan, R. (2021). Penerapan metode HSV pada OpenCV untuk deteksi warna citra digital. *Jurnal Sistem dan Informatika (JAMSI)*, *13*(2), 97–104.

Tarigan, R. (2023). *Penerapan teknik pengolahan citra digital dalam deteksi objek berbasis warna menggunakan OpenCV*. Skripsi. Universitas Negeri Medan.

Pramudyo, S. H. (2022). *Pengolahan Citra Digital: Konsep dan Implementasi dengan Python*. Deepublish.

Azizah, N., & Rahman, F. (2023). *Pengantar Pengolahan Citra Digital*. UMSIDA Press.